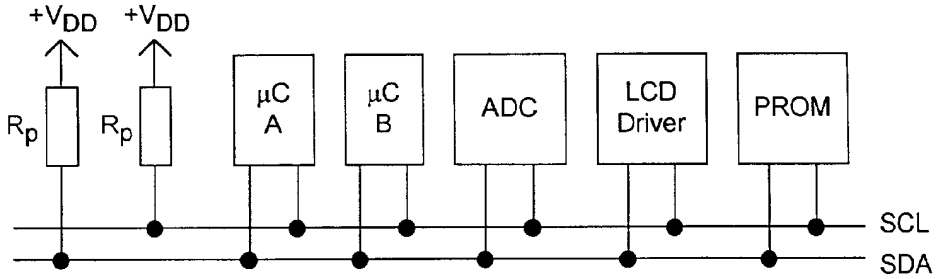
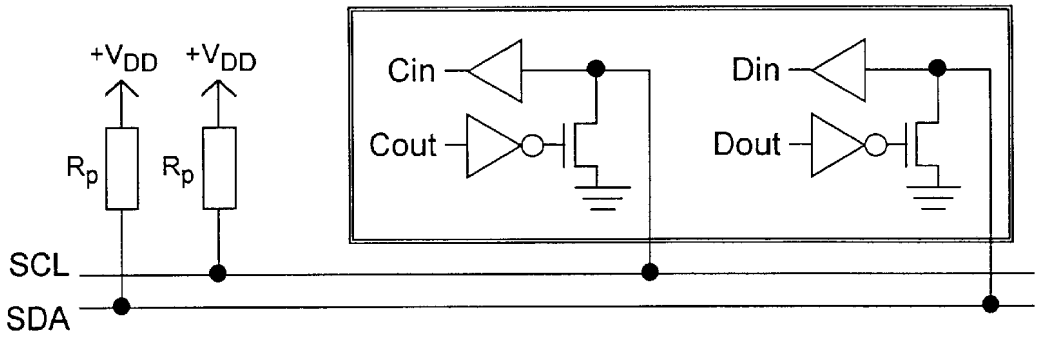
Inter-Integrated Circuit (I^2C) Bus Overview

- Applications
  - Peripheral bus (to single-chip controller devices)
  - Low-cost (simple bus, minimum hardware)
  - Minimum pin resources (two signals)
  - Low speed (serial bus)
- Typical configuration:



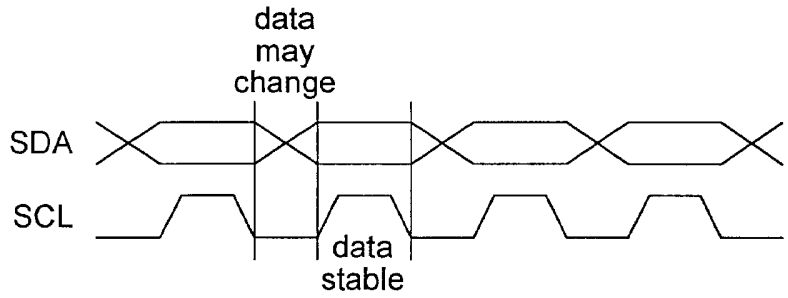
Signalling

- Two bidirectional signals are defined
  - SDA: Serial data
  - SCL: Serial clock
- Electrical interface
  - Devices drive signals using open-collector drivers
  - A pull-up resistor is used on each line
  - Signals are high unless driven low by at least one device (wired-AND)



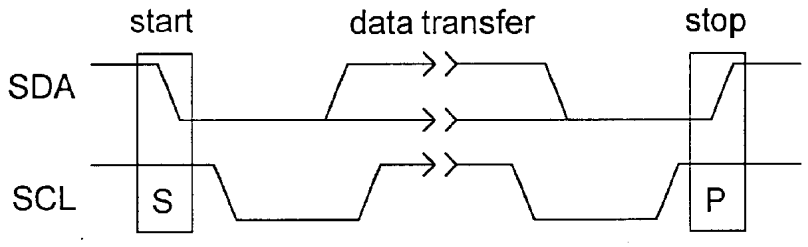
Signalling (continued)

- Bit representation
  - Data value is carried on SDA
  - Logical 1 is V_{DD} (high), but V_{DD} voltage is implementation specific
  - Logical 0 is ground (low)
- Timing
  - SCL provides clock to indicated valid bit
  - Data on SDA must remain valid during the high period of SCL



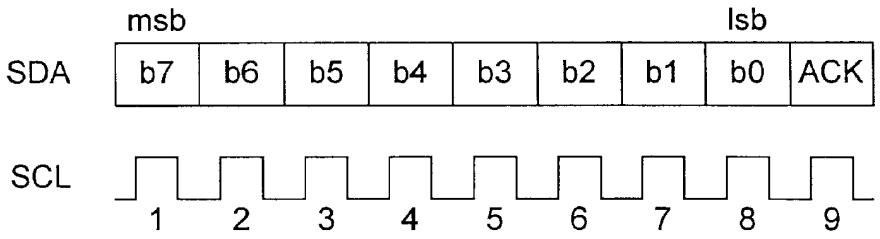
Signalling (continued)

- Framing is needed to mark the beginning and end of a transfer
  - Length fields
  - Special characters
  - Special signal combinations (used on I²C Bus)
- Start condition (S)
  - High-to-low transition on SDA while SCL is high
  - Bus is considered busy after a start condition
  - Re-starts may also occur
- Stop condition (P)
  - Low-to-high transition on SDA while SCL is high
  - Bus is considered released (after a delay) following a stop condition



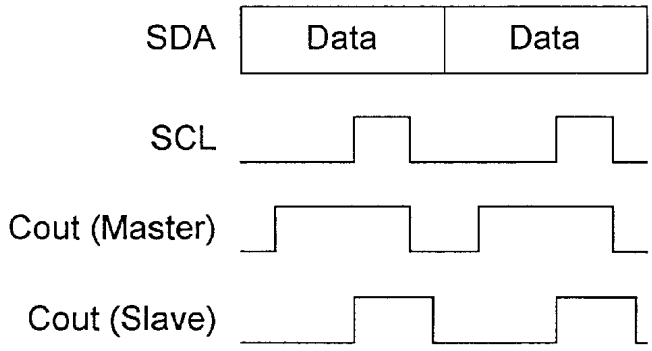
Data Transfer

- Byte format
  - Data transferred in byte (8-bit) units
  - Most significant bit sent first
- Acknowledgment (from receiver)
  - Follows each byte
  - Low is acknowledge, high is negative acknowledge



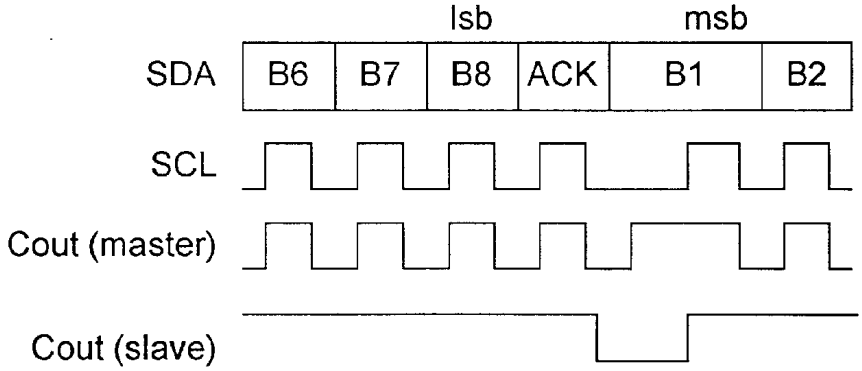
Synchronization

- The wired-AND clock is used for synchronization
  - A bus master generates the clock
  - A bus slave can hold the clock low until it is ready
- Bit-level synchronization
  - Slave device can extend the length of each low clock period



Synchronization (continued)

- Byte-level synchronization
  - Slave devices can hold SCL low after byte received and the acknowledge bit is sent
  - Provides time for slave to process the byte

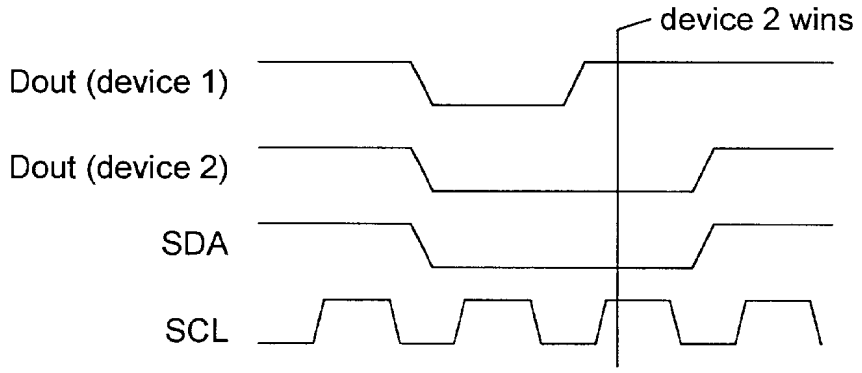


Arbitration

- The I²C Bus allows multiple masters
  - Master: device that initiates a transfer, generates clock signals, and terminates a transfer
  - Slave: device addressed by a master
- Devices may be
  - Only a master
  - Only a slave
  - Sometimes a master and sometimes a slave
- Arbitration ensures that if more than one master attempts to control the bus
  - Only one device is allowed to control the bus
  - Data is not corrupted by the contention
- Basis for arbitration
  - Device cannot attempt to control the bus if another device's start condition has been detected until the controlling master sends a stop condition
  - Wired-AND data line used to detect "collisions"

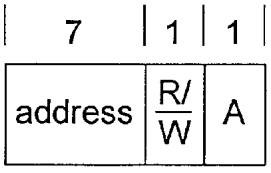
Arbitration (continued)

- Suppose two (or more) masters send a start condition at about the same time
  - Both will attempt to control the bus
- Arbitration takes place on data bus (SDA)
  - As long as both send the same bit, there is no data corruption
  - If one sends a 1 and the other 0, then the 0 is sent (wired-AND)
  - The device sending the 1 loses arbitration



Addressing

- Addresses are seven bits long and uniquely identify a device
  - Addresses are assigned by a standards committee
  - Parts of addresses may be programmed at system reset
    - Permits multiple devices of the same type
- Address is sent by master as the first byte following a start condition
- 8th bit of address selects operation
  - Read (=1): Master receives, slave transmits
  - Write (=0): Master transmits, slave receives
- Addressed slave device responds with an acknowledgment

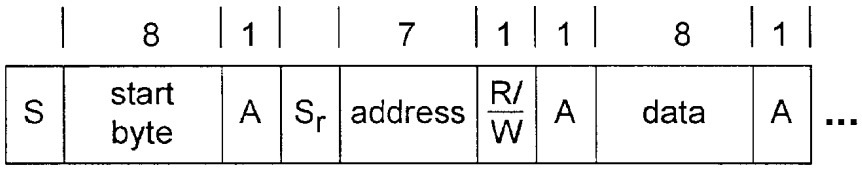


Modes of Operation

- Master transmitter
  - Generates stop and start conditions
  - Generates clock signal
  - Writes data to bus
- Master receiver
  - Generates stop and start conditions
  - Generates clock signal
  - Reads data from bus
  - Generates acknowledge bits
- Slave transmitter
  - Writes data to bus
  - Can "stretch" low clock periods
- Slave receiver
  - Reads data from bus
  - Generates acknowledge bits
  - Can "stretch" low clock periods

Start Procedure for "Low-Speed Mode"

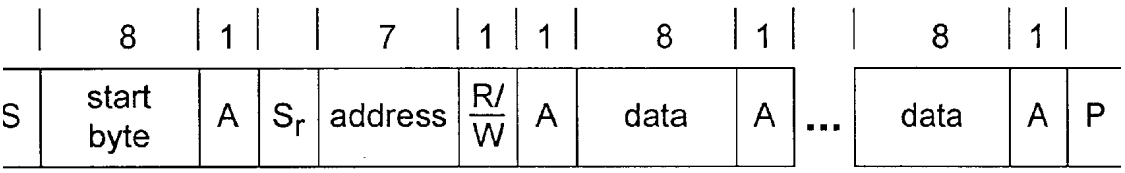
- Slower devices may require extra time to prepare for a transfer
- A special start sequence is used
  - Master sends start condition
  - Master sends start byte (00000001)
  - There is a dummy acknowledge bit
  - Master sends repeated start condition
  - Transaction continues as normal

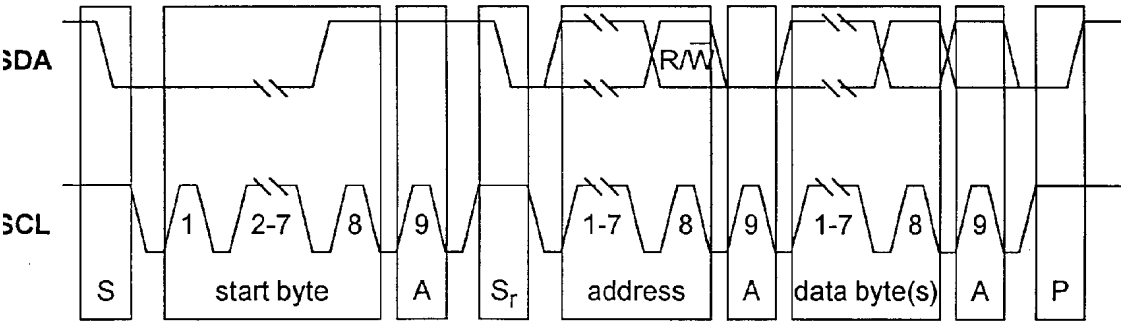


S = start condition
S_r = restart
A = acknowledge

I²C Bus Transaction

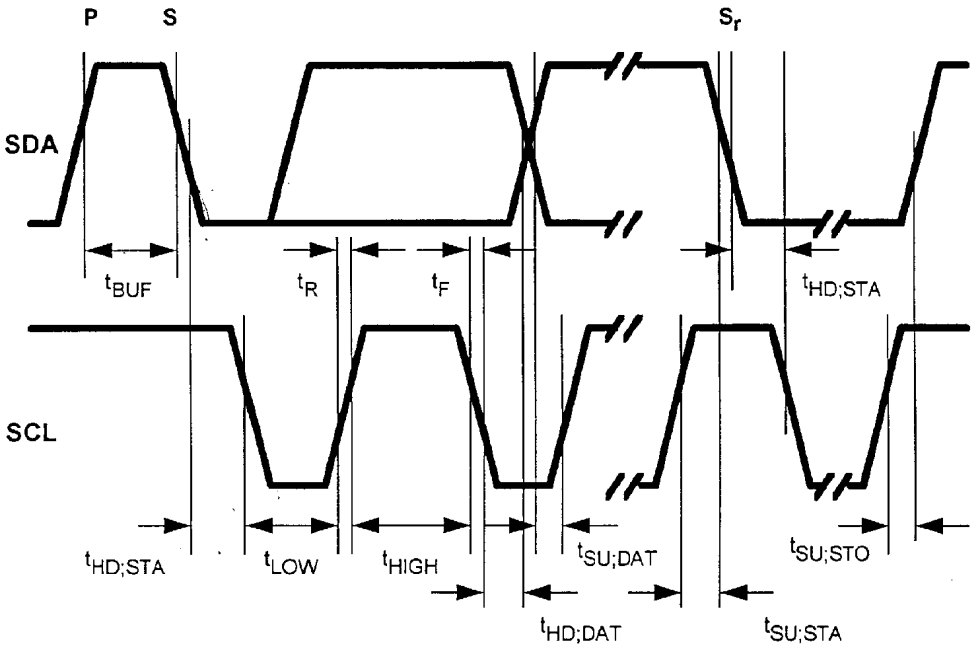
A = acknowledge
P = stop condition





I²C Bus-13

I²C Bus Timing (Slow-Speed Mode)



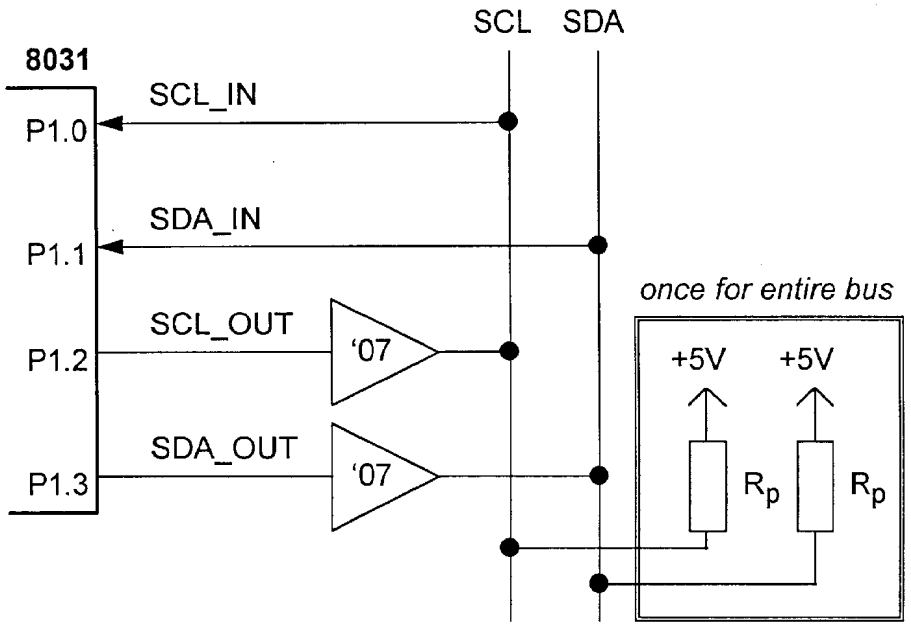
(See *EE 4536 Laboratory Handbook*)

I²C Bus-14

I²C Bus Interface for MCS-51 Microcontrollers

- Hardware controller required for high speed interface
- Low speed interface may be implemented in software using port pins
- SDA and SCL interface using Port 0
  - Port 0 is open-drain, so would be ideal for SDA and SCL
  - Port 0 is not available in expanded mode, an alternative is needed
- Use four pins of Port 1
  - SCL_IN: Sense SCL
  - SCL_OUT: Drive SCL through an open-collector driver (7407)
  - SDA_IN: Sense SDA
  - SDA_OUT: Drive SDA through an open-collector driver

Hardware Interface



Master Transmit Function

- Poll bus to see if free state exists
  - track START conditions from other masters, or
  - monitor for clock activity over maximum idle interval
- Generate START condition on bus
- Send start byte (00000001) and allow dummy acknowledge (=1)
- Generate START condition on bus (re-start)
- Send slave address with write operation (=0)
  - Expect acknowledge from addressed slave (=0)
  - Monitor for bit mis-match for arbitration
- Send data byte(s)
  - Expect acknowledge from addressed slave (=0)
  - Generate clock for data and acknowledge
  - Monitor for bit mis-match for arbitration
- Generate a STOP condition and release bus

Slave Receive Function

- Detect START condition
- Discard start byte and detect START condition (re-start)
- Receive the address byte, compare to device address
  - Receive data only if addressed and write active (=0)
  - Ignore rest of transaction if not addressed
- Generate acknowledge bit (=0) if addressed
- Read data byte(s) from bus
  - Store/process 8 bits of data
  - Generate acknowledge bit (=0)
  - Master generates clock, but slave can stretch low periods
- Detect STOP condition as ending the transaction

Master Receive Function

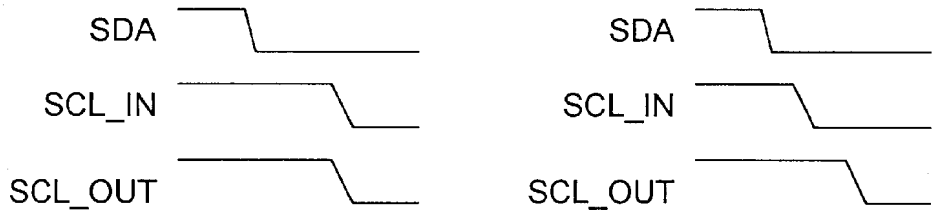
- Poll bus to see if free state exists
  - track START conditions from other masters, or
  - monitor for clock activity over maximum idle interval
- Generate START condition on bus
- Send start byte (00000001) and allow dummy acknowledge (=1)
- Generate START condition on bus (re-start)
- Send slave address with read operation (=1)
  - Expect acknowledge from addressed slave (=0)
  - Monitor for bit mis-match for arbitration
- Receive data byte(s)
  - Generate acknowledge bit (=0)
  - Generate clock for data and acknowledge
  - Monitor for bit mis-match for arbitration
- Generate a STOP condition and release bus
  - Master controls how many bytes are transferred

Slave Transmit Function

- Detect START condition
- Discard start byte and detect START condition (re-start)
- Receive the address byte, compare to device address
  - Transmit data only if addressed and read active (=1)
  - Ignore rest of transaction if not addressed
- Generate acknowledge bit (=0) if addressed
- Transmit data byte(s) from bus
  - Store/process 8 bits of data
  - Receive acknowledge bit from host, stop transmit if negative (=1)
  - Master generates clock, but slave can stretch low periods
- Detect STOP condition as ending the transaction

Example: Send START Condtion

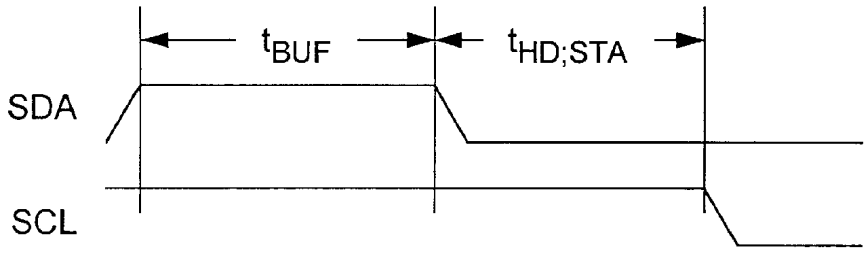
- Master generates a START condition to begin a bus transaction
- Issues
  - Arbitration
  - Signal values (SCL and SDA)
  - Timing
- Arbitration
  - Master attempts to send START only if bus is idle
  - Arbitration may still be lost if another device generates START first
  - Detected by SCL being low before this device drives it low



Arbitration not lost *Arbitration lost*

Example: Send START Condtion (continued)

- Signal values
  - SDA asserted (pulled low)
  - Then, SCL asserted (pulled low)
- Timing
  - Bus must be idle for time t_{BUF} before new transmission can start (not really part of the send START condition function)
  - Hold time of $t_{HD;STA}$ must occur after SDA is asserted before SCL is asserted



Example: Send START Condtion (continued)

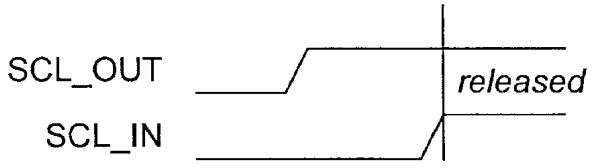
```
------------------------------------------------------------
 SENDS Subroutine (Master function)
   Function: Send start condition on I2C bus
   Outputs:  LOST_ARB FLAG
   Notes:  * Clock must be high on entry for arbitration to
             succeed
           * Exits with clock low if arbitration succeeds
------------------------------------------------------------
ENDS: clr   LOST_ARB        ; Clear lost arbitration flag
      jnb   SCL_IN,sfail    ; Arb fails if clock or data low
      jnb   SDA_IN,sfail

      clr   SDA_OUT         ; SDA := LOW
      mov   R7,#THD_STA     ; Delay for tHD.STA (start hold time)
      lcall DELAY
      jnb   SCL_IN,sfail    ; Arb fails if clock already low
      clr   SCL_OUT         ; SCL := LOW
      ret                   ; Start condition sent, return

fail: setb  SDA_OUT         ; Release data if arbitration fails
      setb  LOST_ARB        ; Set status flag
      ret                   ; Arbitration failed, return
```


Example: Release the Clock

- The clock (SCL) is released (set to high) by either a master or slave after it is asserted (set to low) by that device
- Synchronization
  - This device releases its control of clock (sets SCL_OUT high)
  - Clock is not considered released (SCL_IN high) until all devices release it




```
;-------------------------------------------------
; RELESCL Subroutine
;   Function:  Release clock to high
;-------------------------------------------------

RELESCL: setb  SCL_OUT     ; SCL := HIGH (release
         jnb   SCL_IN,*    ; Wait for all to release
         ret               ; Clock released to high
```